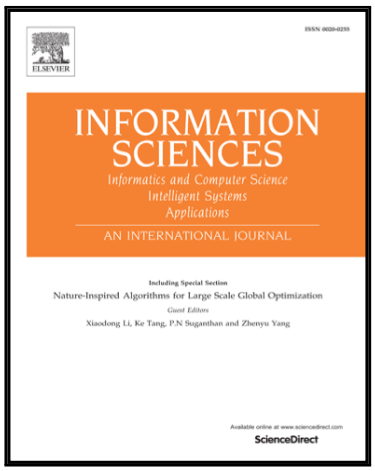

Online Semi-Supervised Support Vector Machine

Ying Liu, Zhen Xu, Chunguang Li

PII: S0020-0255(18)30056-2
DOI: 10.1016/j.ins.2018.01.048
Reference: INS 13402

To appear in: *Information Sciences*

Received date: 25 February 2017
Revised date: 21 January 2018
Accepted date: 24 January 2018

Please cite this article as: Ying Liu, Zhen Xu, Chunguang Li, Online Semi-Supervised Support Vector Machine, *Information Sciences* (2018), doi: 10.1016/j.ins.2018.01.048

Online Semi-Supervised Support Vector Machine

Ying Liu, Zhen Xu, Chunguang Li*

College of Information Science and Electronic Engineering
Zhejiang University, Hangzhou 310027, P.R. China.

Abstract

Recently, support vector machine (SVM) has received much attention due to its good performance and wide applicability. As a supervised learning algorithm, the standard SVM uses sufficient labeled data to obtain the optimal decision hyperplane. However, in many practical applications, it is difficult and/or expensive to obtain labeled data. Besides, the standard SVM is a batch learning algorithm. It is inefficient to handle streaming data as the classifier must be retrained from scratch whenever a new data is arrived. In this paper, we consider the online classification of streaming data when only a small portion of data are labeled while a large portion of data are unlabeled. In order to obtain an adaptive solution with relatively low computational complexity, a new form of manifold regularization is proposed. Then, an adaptive and online semi-supervised least square SVM is developed, which well exploits the information of new incoming labeled or unlabeled data to boost learning performance. Simulations on synthetic and real data sets show that the proposed algorithm achieves good classification performance even if there only exist a few labeled data.

Keywords: support vector machine, semi-supervised learning, online learning, classification, least-square SVM, manifold regularization

1. Introduction

The standard support vector machine (SVM) is a well-known supervised machine learning algorithm, which has been widely used for data classification [15, 18, 21, 26]. It seeks the optimal decision hyperplane using the principle of structural risk minimization. Extensive studies have shown that the standard SVM can obtain good performance on various classification problems. But, as a supervised learning algorithm, sufficient labeled data are needed in order to achieve good classification performance.

As we know, it is difficult and/or costly to collect a large amount of labeled data in many applications, especially in the field of biology and medicine. However, unlabeled data are relatively easy and cheap to obtain. Considering this, semi-supervised learning

*Corresponding author.
Email addresses: yingliu@zju.edu.cn (Ying Liu), 414934116@qq.com (Zhen Xu), cgli@zju.edu.cn (Chunguang Li)

(SSL), which employs a small portion of labeled data and a large portion of unlabeled data, has attracted much attention [2, 8, 20, 24]. By exploiting potential information of both unlabeled data and labeled data, the SSL can significantly improve the learning performance [4, 25, 30]. So, SSL has been widely used in many areas, such as text classification [16], image recognition [28, 29] and natural language processing [23].

Recently, the concept of SSL has been incorporated into the SVM, and some semi-supervised SVM (S^3VM) algorithms have been proposed [3, 12]. It has shown in the literatures that S^3VM algorithms can outperform the standard SVM, if the valuable information underlying the unlabeled data can be adequately used. Yet, most of the existing S^3VM algorithms are batch learning algorithms, in which the whole training data must be stored in advance and computed in each training process. So, they are inefficient to handle large-scale data set due to the huge space and time requirements. Besides, in many practical applications, the training data are sequentially generated even at an unprecedented rate. In these cases, once a new sample is added to the training set, the classifier must be retrained from scratch, which is time-consuming.

To tackle this problem, in [27], an online semi-supervised least square SVM (S^2LS-SVM) algorithm using the standard manifold regularization has been proposed. This method obtains the optimal solution by solving the linear algebraic equations (LAEs), which greatly simplifies the training process. However, in this algorithm, in order to derive an incremental solution, the parameter matrix of the LAEs satisfying the Karush-Kuhn-Tucker (KKT) conditions is (over) simplified. Besides, the information of unlabeled data is not fully exploited. Thus, the solution is rather coarse, and its performance is unsatisfied in some cases, especially when the initial number of labeled data is small.

In this paper, some improvements on S^2LS-SVM are made from several aspects, and an online S^3VM (OS^3VM) algorithm using a new form of manifold regularization is proposed. The proposed OS^3VM can not only handle the new incoming data effectively, but also control the computational complexity at an acceptable level. The main contributions of this paper are summarized as follows.

- Different from the previous work [27], not only the information of labeled data but also that of unlabeled data is fully considered in both initialization and the latter iteration process to boost the classification performance.
- In order to obtain a more accurate adaptive solution, the effect of new incoming data on manifold regularization is well considered. This significantly increases the difficulty in the online implementation, which makes it non-trivial. To tackle this problem, we propose a new form of manifold regularization. Moreover, we present a method to decompose the high-dimensional incremental matrix of manifold regularization into low-dimensional matrices so as to avoid expensive computation of matrix inversion.

The rest of the paper is organized as follows. In Section 2, the problem of classification is formulated, and some related works of S^2LS-SVM are introduced. In Section 3, the problems incurred when S^2LS-SVM is directly extended to the online implementation are discussed. In Section 4, a new form of manifold regularization

is proposed and then the OS^{3}VM algorithm is developed. Experimental results are provided in Section 5. Finally, some conclusions are drawn in Section 6.

Notations: In this paper, the superscript $(\cdot)^T$ denotes transposition, and the notation $\{\cdot\}$ denotes a set. The notation $[\cdot]_i$ denotes the i-th entry of a vector and $[\cdot]_{ij}$ denotes the ij-th entry of a matrix. The notations $\mathbf{1}$ and $\mathbf{0}$ represent the vectors with all coefficients being 1 and 0, respectively. The notations $(\cdot)^{\text{UL}}$, $(\cdot)^{\text{UR}}$, $(\cdot)^{\text{LL}}$ and $(\cdot)^{\text{LR}}$ denote the upper-left, upper-right, lower-left and lower-right sub-blocks of a matrix, respectively. The notation $k(\cdot,\cdot)$ denotes a kernel function, and K denotes a matrix with each element computed by the kernel function $k(\cdot,\cdot)$. Additionally, $D(\cdot)$ denotes a diagonal matrix. Other notations will be introduced if necessary.

2. Problem Formulation and Related Works

In this section, the problem of classification using SSL is formulated, and then the batch S^{2}LS-SVM algorithm proposed in [27] is briefly reviewed.

2.1. Problem Formulation

We consider a binary classification problem with l labeled data $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_l, y_l)\}$ and u unlabeled data $\{\boldsymbol{x}_{l+1}, \ldots, \boldsymbol{x}_{l+u}\}$, where $\boldsymbol{x}_i \in X \subseteq \mathbb{R}^m$ denotes the i-th input vector (a vector of attributes), and $y_i \in \{-1, 1\}$ denotes its corresponding class label. Our objective is to find the optimal classification hyperplane between two classes of samples such that the data can be classified with high accuracy.

Here, we assume that the discriminant function for classification is nonlinear, which can be expressed as an expansion of kernel functions associated with labeled and unlabeled data. Then, the predicted output of the discriminant function with respect to the i-th input vector $\boldsymbol{x}_i$ can be expressed as

$$f(\boldsymbol{x}_i) = \sum_{k=1}^{l+u} \alpha_k k(\boldsymbol{x}_i, \boldsymbol{x}_k) + b, \tag{1}$$

where b denotes the bias term, $k(\boldsymbol{x}_i, \boldsymbol{x}_k) =< \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_k) >$ denotes the kernel function for a data pair $(\boldsymbol{x}_i, \boldsymbol{x}_k)$ and $\phi(\cdot)$ denotes a nonlinear mapping to the reproducing kernel Hilbert space (RKHS), and α_k denotes the corresponding weight. For the considered two-class classification problem, once the discriminant function is obtained, we have the class label $y_i = \text{sgn}(f(\boldsymbol{x}_i))$ for the i-th sample.

In addition, to carry out the following study, the smoothness assumption generally used in the context of SSL [3, 12, 14, 20], is given.

Smoothness Assumption: If two samples $\boldsymbol{x}_i$, $\boldsymbol{x}_j$ in the high-density regions (the manifold) in the geodesic distance are close, then so should be their corresponding labels.

2.2. Related Works of S^2LS-SVM

In [27], the S^2LS-SVM algorithm has been proposed, which is derived by solving the following optimization problem

$$\begin{aligned} \min_{f} \quad & \frac{C}{2}\sum_{i=1}^{l} e_i^2 + \frac{1}{2}\gamma_{\mathcal{H}}\|f\|_{\mathcal{H}}^2 + \frac{1}{2}\gamma_{\mathcal{M}}\|f\|_{\mathcal{M}}^2 \\ \text{s.t.} \quad & y_i(\sum_{j=1}^{l+u}\alpha_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) + b) = 1 - e_i, \quad i = 1, \ldots, l, \end{aligned} \tag{2}$$

where e_i denotes the prediction error for the i-th labeled data, and $C > 0$ denotes the corresponding weight parameter.

The term $\|f\|_{\mathcal{H}}^2$ in (2) is the norm of the discriminant function in the RKHS regularized by the weight parameter $\gamma_{\mathcal{H}}$ to prevent f being too wiggly outside the training samples, which is given by

$$\|f\|_{\mathcal{H}}^2 = \boldsymbol{\alpha}^T K \boldsymbol{\alpha}, \tag{3}$$

where $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_{l+u}]^T$, and K denotes a $(l+u) \times (l+u)$ matrix with each element computed by the kernel function.

In (2), the manifold regularization $\|f\|_{\mathcal{M}}^2$ with weight parameter $\gamma_{\mathcal{M}}$ is used to guarantee that the labels vary smoothly along the graph represented by graph Laplacian [14], given by

$$\begin{aligned} \|f\|_{\mathcal{M}}^2 &= \sum_{i=1}^{l+u}\sum_{j=1}^{l+u} W_{ij}(f(\boldsymbol{x}_i) - f(\boldsymbol{x}_j))^2 \\ &= \boldsymbol{f}^T L \boldsymbol{f}, \end{aligned} \tag{4}$$

where $\boldsymbol{f} = [f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_{l+u})]^T$. The graph Laplacian matrix $L = D(\boldsymbol{d}) - W$ depends on both labeled and unlabeled data, where W is a weight matrix with each element $W_{ij} > 0$ measuring the similarity of a data pair $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and being calculated by Gaussian kernel function, and $D(\boldsymbol{d}) = \text{diag}\{d_1, \ldots, d_{l+u}\}$ is a diagonal matrix with each diagonal element $d_i = \sum_{j=1}^{l+u} W_{ij}$.

Substituting (1) into (4), $\|f\|_{\mathcal{M}}^2$ becomes

$$\|f\|_{\mathcal{M}}^2 = \boldsymbol{\alpha}^T KLK\boldsymbol{\alpha}. \tag{5}$$

Substituting (3) and (5) into (2) and introducing a multiplier vector $\boldsymbol{\beta}$, the Lagrangian (2) associated to the optimization problem becomes

$$\begin{aligned} \mathcal{L}(\boldsymbol{e}, b, \boldsymbol{\beta}, \boldsymbol{\alpha}) = & \frac{C}{2}\boldsymbol{e}^T\boldsymbol{e} + \frac{1}{2}\boldsymbol{\alpha}^T(\gamma_{\mathcal{H}}K + \gamma_{\mathcal{M}}KLK)\boldsymbol{\alpha} \\ & - \boldsymbol{\beta}^T\left(D(\boldsymbol{y})(JK\boldsymbol{\alpha} + b\mathbf{1}) - \mathbf{1} + \boldsymbol{e}\right), \end{aligned} \tag{6}$$

where $\boldsymbol{e} = [e_1, \ldots, e_l]^T$, $D(\boldsymbol{y}) = \text{diag}\{y_1, \ldots, y_l\}$, and $J = [I_{l\times l} \quad \mathbf{0}_{l\times u}]$ is a $l \times (l+u)$ matrix with $I_{l\times l}$ being a $l \times l$ identity matrix and $\mathbf{0}_{l\times u}$ being a $l \times u$ matrix consisting of all zero entities.

Then, by setting the gradient of the objective function with respect to $\boldsymbol{e}, b, \boldsymbol{\beta}, \boldsymbol{\alpha}$ to zero, we have

$$\begin{cases} \dfrac{\partial \mathcal{L}}{\partial \boldsymbol{e}} = C\boldsymbol{e} - \boldsymbol{\beta} = \mathbf{0}, \\ \dfrac{\partial \mathcal{L}}{\partial b} = \boldsymbol{\beta}^T D(\boldsymbol{y})\mathbf{1} = 0, \\ \dfrac{\partial \mathcal{L}}{\partial \boldsymbol{\beta}} = D(\boldsymbol{y})(JK\boldsymbol{\alpha} + b\mathbf{1}) - \mathbf{1} + \boldsymbol{e} = \mathbf{0}, \\ \dfrac{\partial \mathcal{L}}{\partial \boldsymbol{\alpha}} = P\boldsymbol{\alpha} - KJ^T D(\boldsymbol{y})\boldsymbol{\beta} = \mathbf{0}. \end{cases} \tag{7}$$

where

$$P = \gamma_{\mathcal{H}} K + \gamma_{\mathcal{M}} KLK. \tag{8}$$

Then, the LAEs satisfying the KKT conditions become

$$\Psi\lambda = Y, \tag{9}$$

where

$$\Psi = \begin{bmatrix} 0 & \boldsymbol{y}^T & \mathbf{0}_{1\times(l+u)} \\ \boldsymbol{y} & \frac{1}{C} I_{l\times l} & D(\boldsymbol{y})JK \\ \mathbf{0}_{1\times(l+u)} & -KJ^T D(\boldsymbol{y}) & P \end{bmatrix}, \tag{10}$$

and

$$\lambda = \begin{bmatrix} b & \boldsymbol{\beta}^T & \boldsymbol{\alpha}^T \end{bmatrix}^T, \quad Y = \begin{bmatrix} 0 & \mathbf{1}_{1\times l} & \mathbf{0}_{1\times(l+u)} \end{bmatrix}^T.$$

The optimal $\boldsymbol{\alpha}$ and b can be obtained by solving $\lambda = \Psi^{-1}Y$. Consequently, the optimal discriminant function f can be determined, and thus the data can be classified.

3. Problems in the Online Implementation of S^{2}LS-SVM

In this section, we are interested in the online classification of streaming data. It is obvious that the batch S^{2}LS-SVM is not applicable to these cases as the classifier must be retrained whenever a new data is added to or removed from the current training data set. Considering this, in this section, we try to extend the batch S^{2}LS-SVM based on the standard manifold regularization to the online version, and then discuss the problems that brings in.

The basic settings of the online classification problem are similar to those illustrated in Sec. 2.1, except that the data (labeled or unlabeled) are added to the current training data set one by one. Our target is to provide a good discriminant function through efficient online learning such that the data can be classified with high accuracy. It is emphasized that to differentiate the online S^{2}LS-SVM from the batch S^{2}LS-SVM, the quantities for online version are denoted by the notations in the form of $(\cdot)_t$ to enforce the time.

At time instant t, we assume that the training set contains total l_t labeled data and u_t unlabeled data. Then, the instantaneous LAEs satisfying the KKT condition (9) become

$$\Psi_t \lambda_t = Y_t, \tag{11}$$

where

$$\lambda_t = \begin{bmatrix} b_t & \boldsymbol{\beta}_t^T & \boldsymbol{\alpha}_t^T \end{bmatrix}^T, \qquad Y_t = \begin{bmatrix} 0 & \mathbf{1}_{1\times l_t} & \mathbf{0}_{1\times(l_t+u_t)} \end{bmatrix}^T, \tag{12}$$

and

$$\Psi_t = \begin{bmatrix} 0 & \boldsymbol{y}_t^T & \mathbf{0}_{1\times(l_t+u_t)} \\ \boldsymbol{y}_t & \frac{1}{C} I_{l_t\times l_t} & D(\boldsymbol{y}_t) J_t K_t \\ \mathbf{0}_{(l_t+u_t)\times 1} & -K_t J_t^T D(\boldsymbol{y}_t) & P_t \end{bmatrix}, \tag{13}$$

with the instantaneous sub-block

$$P_t = \gamma_{\mathcal{H}} K_t + \gamma_{\mathcal{M}} K_t L_t K_t. \tag{14}$$

It is noted that Ψ_t is a $(2l_t + u_t + 1) \times (2l_t + u_t + 1)$ matrix, where P_t, K_t and L_t (corresponding to K and L in the batch S^{2}LS-SVM respectively) are $(l_t + u_t) \times (l_t + u_t)$ matrices, $\boldsymbol{y}_t$ is a l_t-dimensional column vector, and $J_t = [I_{l_t\times l_t} \quad \mathbf{0}_{l_t\times u_t}]$. Both λ_t and Y_t are $(2l_t + u_t + 1)$-dimensional column vectors.

At the next time $t+1$, whenever a new data is arrived, based on the KKT conditions, the LAEs must be updated. According to the types of data (whether it is labeled or unlabeled), there are two different forms of LAEs. Nevertheless, in order to show the incremental relationship between the last Ψ_t and the current Ψ_{t+1} clearly, we first give the general form of LAEs for these two cases. At time $t + 1$, the LAEs satisfying the KKT conditions can be uniformly expressed as

$$\Psi_{t+1} \lambda_{t+1} = Y_{t+1}, \tag{15}$$

where

$$\Psi_{t+1} = \begin{bmatrix} 0 & \boldsymbol{y}_{t+1}^T & \mathbf{0}_{1\times(l_{t+1}+u_{t+1})} \\ \boldsymbol{y}_{t+1} & \frac{1}{C} I_{l_{t+1}\times l_{t+1}} & D(\boldsymbol{y}_{t+1}) J_{t+1} K_{t+1} \\ \mathbf{0}_{(l_{t+1}+u_{t+1})\times 1} & -K_{t+1} J_{t+1}^T D(\boldsymbol{y}_{t+1}) & P_{t+1} \end{bmatrix},$$

and

$$\lambda_{t+1} = \begin{bmatrix} b_{t+1} & \boldsymbol{\beta}_{t+1}^T & \boldsymbol{\alpha}_{t+1}^T \end{bmatrix}^T, \quad Y_{t+1} = \begin{bmatrix} 0 & \mathbf{1}_{1\times l_{t+1}} & \mathbf{0}_{1\times(l_{t+1}+u_{t+1})} \end{bmatrix}^T.$$

Next, we would like to relate the matrices K_{t+1}, L_{t+1} and P_{t+1} in current Ψ_{t+1} to their values in last Ψ_t. Whenever a new data is added to the current training set, both K_{t+1} and L_{t+1} increase by one dimension. According to their definitions, we have

$$K_{t+1} = \begin{bmatrix} K_t & \boldsymbol{k}_{t+1} \\ \boldsymbol{k}_{t+1}^T & 1 \end{bmatrix}, \text{ and } L_{t+1} = \begin{bmatrix} L_t + D(\boldsymbol{k}_{t+1}) & -\boldsymbol{k}_{t+1} \\ -\boldsymbol{k}_{t+1}^T & d_{t+1} \end{bmatrix}, \tag{16}$$

where $\boldsymbol{k}_{t+1} = [k(\boldsymbol{x}_1, \boldsymbol{x}_{t+1}), \ldots, k(\boldsymbol{x}_t, \boldsymbol{x}_{t+1})]^T$, and $d_{t+1} = \mathbf{1}^T \boldsymbol{k}_{t+1}$.

Then, based on K_{t+1} and L_{t+1}, the term P_{t+1} becomes

$$P_{t+1} = \gamma_{\mathcal{H}} K_{t+1} + \gamma_{\mathcal{M}} K_{t+1} L_{t+1} K_{t+1}. \tag{17}$$

In order to show the incremental relationship between P_t and P_{t+1} conveniently, we decompose matrix P_{t+1} into four sub-blocks, i.e.

$$P_{t+1} \triangleq \begin{bmatrix} P_{t+1}^{\mathrm{UL}} & P_{t+1}^{\mathrm{UR}} \\ P_{t+1}^{\mathrm{LL}} & P_{t+1}^{\mathrm{LR}} \end{bmatrix},$$

where

$$\begin{aligned} P_{t+1}^{\mathrm{UL}} &= (\gamma_{\mathcal{H}} K_t + \gamma_{\mathcal{M}} K_t L_t K_t) + \gamma_{\mathcal{M}}[K_t D(\boldsymbol{k}_{t+1}) K_t - \boldsymbol{k}_{t+1}\boldsymbol{k}_{t+1}^T K_t - K_t \boldsymbol{k}_{t+1}\boldsymbol{k}_{t+1}^T + d_{t+1}\boldsymbol{k}_{t+1}\boldsymbol{k}_{t+1}^T] \in \mathbb{R}^{(l_t+u_t)\times(l_t+u_t)}, \\ &\triangleq P_t + \triangle P_t, \\ P_{t+1}^{\mathrm{UR}} &= \gamma_{\mathcal{H}} \boldsymbol{k}_{t+1} + \gamma_{\mathcal{M}}[K_t(L_t + D(\boldsymbol{k}_{t+1}))\boldsymbol{k}_{t+1} - \boldsymbol{k}_{t+1}\boldsymbol{k}_{t+1}^T \boldsymbol{k}_{t+1} - K_t \boldsymbol{k}_{t+1} + d_{t+1}\boldsymbol{k}_{t+1}] \in \mathbb{R}^{(l_t+u_t)\times 1}, \\ P_{t+1}^{\mathrm{LL}} &= \gamma_{\mathcal{H}} \boldsymbol{k}_{t+1}^T + \gamma_{\mathcal{M}}[\boldsymbol{k}_{t+1}^T(L_t + D(\boldsymbol{k}_{t+1}))K_t - \boldsymbol{k}_{t+1}^T K_t + (-\boldsymbol{k}_{t+1}^T \boldsymbol{k}_{t+1} + d_{t+1})\boldsymbol{k}_{t+1}^T] \in \mathbb{R}^{1\times(l_t+u_t)}, \\ P_{t+1}^{\mathrm{LR}} &= \gamma_{\mathcal{H}} + \gamma_{\mathcal{M}}[\boldsymbol{k}_{t+1}^T(L_t + D(\boldsymbol{k}_{t+1}))\boldsymbol{k}_{t+1} - 2\boldsymbol{k}_{t+1}^T \boldsymbol{k}_{t+1} + d_{t+1}] \in \mathbb{R}^{1\times 1}, \end{aligned}$$

and $\triangle P_t$ in P_{t+1}^{UL} being

$$\triangle P_t = \gamma_{\mathcal{M}}[K_t D(\boldsymbol{k}_{t+1}) K_t - \boldsymbol{k}_{t+1}\boldsymbol{k}_{t+1}^T K_t - K_t \boldsymbol{k}_{t+1}\boldsymbol{k}_{t+1}^T + d_{t+1}\boldsymbol{k}_{t+1}\boldsymbol{k}_{t+1}^T] \in \mathbb{R}^{(l_t+u_t)\times(l_t+u_t)}. \tag{18}$$

Remark 1: From (14), we can see that P_t contains the product of matrices K_t and L_t. At time $t+1$, when some new elements are added to K_{t+1} and L_{t+1} (see (16)), the upper left (UL) sub-block P_{t+1}^{UL} changes from P_t, due to matrix multiplication between the new K_{t+1} and L_{t+1}. Here, we use $\triangle P_t$ to denote the increment caused by the new added data for simplification, which is given in (18). In S^{2}LS-SVM algorithm, in order to simplify the computation, the term $\triangle P_t$ is ignored, and thus the performance of classification is affected [27]. In this paper, the effect of $\triangle P_t$ is fully considered to improve the classification performance. But, with such a consideration, the explicit computation of high-dimensional matrix inversion cannot be avoided, if the standard manifold regularization is used. The details will be discussed latter in this section.

As mentioned above, Ψ_{t+1} depends on whether the new incoming data is labeled or unlabeled. Next, we will give the exact expressions of Ψ_{t+1} for these two cases.

Case 1: The incoming data is unlabeled.

At time $t+1$, if the new incoming data $\boldsymbol{x}_{t+1}$ is unlabeled, the total number of unlabeled data in the training data set increases by 1, i.e. $u_{t+1} = u_t + 1$, while that of labeled data keeps invariant, i.e. $l_{t+1} = l_t$. According to (2), the new incoming unlabeled data affects both the regularization term $\|f\|_{\mathcal{H}}^2$ and manifold regularization term $\|f\|_{\mathcal{M}}^2$. Consequently, the parameter vector $\boldsymbol{\alpha}_{t+1}$ in λ_{t+1} increases by one dimension comparing to $\boldsymbol{\alpha}_t$, while the dimensions of both b_{t+1} and $\boldsymbol{\beta}_{t+1}$ in λ_{t+1} keep invariant. Correspondingly, both Ψ_{t+1} and Y_{t+1} increase by one dimension, which are expressed as

$$\begin{aligned} \Psi_{t+1} &= \left[\begin{array}{ccc|c} 0 & \boldsymbol{y}_t & \mathbf{0}_{1\times(l_t+u_t)} & 0 \\ \boldsymbol{y}_t & \frac{1}{C} I_{l_t\times l_t} & D(\boldsymbol{y}_t) J_t K_t & D(\boldsymbol{y}_t) J_t \boldsymbol{k}_{t+1} \\ \mathbf{0}_{(l_t+u_t)\times 1} & -K_t J_t^T D(\boldsymbol{y}_t) & P_{t+1}^{\mathrm{UL}} & P_{t+1}^{\mathrm{UR}} \\ \hline 0 & -\boldsymbol{k}_{t+1}^T J_t^T D(\boldsymbol{y}_t) & P_{t+1}^{\mathrm{LL}} & P_{t+1}^{\mathrm{LR}} \end{array}\right] \\ &\triangleq \left[\begin{array}{c|c} \Psi_{t+1}^{\mathrm{UL}} & \Psi_{t+1}^{\mathrm{UR}} \\ \hline \Psi_{t+1}^{\mathrm{LL}} & \Psi_{t+1}^{\mathrm{LR}} \end{array}\right], \\ Y_{t+1} &= \left[\begin{array}{c|c} Y_t^T & 0 \end{array}\right]^T. \end{aligned} \tag{19}$$

Note that in (19), for clarity, those elements introduced by the new incoming data are colored in blue. As to the term P^{UL}_{t+1}, since it is changed from P_t when a new data is added, it is colored in red. Other elements that keep invariant as their last values at time instant t are denoted in black. It is noted that the similar notations are used in the following Case 2 without specific explanation. Besides, in order to show the differences between the matrices Ψ_t and Ψ_{t+1}, we decompose the matrix Ψ_{t+1} into four sub-blocks Ψ^{UL}_{t+1}, Ψ^{UR}_{t+1}, Ψ^{LL}_{t+1} and Ψ^{LR}_{t+1} respectively, which are divided by the dotted line.

Case 2: The incoming data is labeled.

If the new incoming labeled data $\boldsymbol{x}_{t+1}$ is labeled with label y_{t+1}, we have $u_{t+1} = u_t$, $l_{t+1} = l_t + 1$. Different from Case 1, the new added data not only has impact on the regularization terms $\|f\|^2_{\mathcal{H}}$ and $\|f\|^2_{\mathcal{M}}$, but also brings in an additional constraint equation. Consequently, the current parameter vectors $\boldsymbol{\alpha}_{t+1}$ and $\boldsymbol{\beta}_{t+1}$ in λ_{t+1} satisfying the KKT conditions both increase by one dimension. Correspondingly, Ψ_{t+1} and Y_{t+1} increase by two dimensions, which are given by

$$\Psi_{t+1}=\begin{bmatrix} 0 & \boldsymbol{y}_t & y_{t+1} & \mathbf{0}_{1\times(l_t+u_t)} & 0 \\ \boldsymbol{y}_t & \frac{1}{C}I_{l_t\times l_t} & \mathbf{0}_{l_t\times 1} & D(\boldsymbol{y}_t)J_tK_t & D(\boldsymbol{y}_t)J_t\boldsymbol{k}_{t+1} \\ y_{t+1} & \mathbf{0}_{1\times l_t} & \frac{1}{C} & y_{t+1}\boldsymbol{k}^T_{t+1} & y_{t+1} \\ \mathbf{0}_{(l_t+u_t)\times 1} & -K_tJ_t^TD(\boldsymbol{y}_t) & -y_{t+1}\boldsymbol{k}_{t+1} & P^{\mathrm{UL}}_{t+1} & P^{\mathrm{UR}}_{t+1} \\ 0 & -\boldsymbol{k}^T_{t+1}J_t^TD(\boldsymbol{y}_t) & -y_{t+1} & P^{\mathrm{LL}}_{t+1} & P^{\mathrm{LR}}_{t+1} \end{bmatrix}, \tag{20}$$
$$Y_{t+1}=\begin{bmatrix} 0 & \mathbf{1}_{1\times l_t} & 1 & \mathbf{0}_{1\times(l_t+u_t)} & 0 \end{bmatrix}^T.$$

To give a similar form as (19) of Case 1, where the new added elements lie in the UR, LL and LR sub-blocks, we reorder the rows and the columns of Ψ_{t+1} and Y_{t+1} such that

$$E_t\Psi_{t+1}E_t=\left[\begin{array}{ccc:cc} 0 & \boldsymbol{y}_t & \mathbf{0}_{1\times(l_t+u_t)} & y_{t+1} & 0 \\ \boldsymbol{y}_t & \frac{1}{C}I_{l_t\times l_t} & D(\boldsymbol{y}_t)J_tK_t & \mathbf{0}_{l_t\times 1} & D(\boldsymbol{y}_t)J_t\boldsymbol{k}_{t+1} \\ \mathbf{0}_{(l_t+u_t)\times 1} & -K_tJ_t^TD(\boldsymbol{y}_t) & P^{\mathrm{UL}}_{t+1} & -y_{t+1}\boldsymbol{k}_{t+1} & P^{\mathrm{UR}}_{t+1} \\ \hdashline y_{t+1} & \mathbf{0}_{1\times l_t} & y_{t+1}\boldsymbol{k}^T_{t+1} & \frac{1}{C} & y_{t+1} \\ 0 & -\boldsymbol{k}^T_{t+1}J_t^TD(\boldsymbol{y}_t) & P^{\mathrm{LL}}_{t+1} & -y_{t+1} & P^{\mathrm{LR}}_{t+1} \end{array}\right] \tag{21}$$
$$\triangleq\left[\begin{array}{c:c} \Psi^{\mathrm{UL}}_{t+1} & \Psi^{\mathrm{UR}}_{t+1} \\ \hdashline \Psi^{\mathrm{LL}}_{t+1} & \Psi^{\mathrm{LR}}_{t+1} \end{array}\right],$$
$$E_tY_{t+1}=\left[\begin{array}{ccc:cc} 0 & \mathbf{1}_{1\times l_t} & \mathbf{0}_{1\times(l_t+u_t)} & 1 & 0 \end{array}\right]^T=\left[\begin{array}{c:c} Y_t^T & \begin{array}{cc} 1 & 0 \end{array} \end{array}\right]^T.$$

In (21), $E_t = E(2l_t + u_t + 1, 2l_t + u_t + 2)\ldots E(l_t + 2, l_t + 3)$ is the product of a series of elementary matrices $E(i, j) = I - (\boldsymbol{\xi}_i - \boldsymbol{\xi}_j)(\boldsymbol{\xi}_i - \boldsymbol{\xi}_j)^T$, where $\boldsymbol{\xi}_i$ is a column vector with only the i-th element being 1, while the others being 0. Similar to Case 1, we also decompose the reordered matrix $E_t\Psi_{t+1}E_t$ into four sub-blocks as separated by the dotted line.

Based on Ψ_{t+1} and Y_{t+1}, the optimal parameter vector λ_{t+1} for both Case 1 and Case 2 can be obtained by

$$\lambda_{t+1} = \Psi^{-1}_{t+1}Y_{t+1}. \tag{22}$$

For the online implementation, with the increasing number of the data, it is costly to solve Ψ^{-1}_{t+1} directly because of the complex computation of high-dimensional matrix inversion. To tackle this problem, we can decompose Ψ_{t+1} or reordered Ψ_{t+1} into some sub-blocks such that some sub-blocks have already known at time $t + 1$, while the

others can be easily computed [11, 17]. If it is feasible, an incremental solution of Ψ_{t+1}^{-1} can be obtained without explicitly computing the high-dimensional matrix inverse. The decomposition results as separated by the dotted line are given in (19) and (21) for Case 1 and Case 2, respectively.

Then, according to the block matrix inversion lemma [9, 10], we have

$$\Xi\Psi_{t+1}^{-1}\Xi=\begin{bmatrix}(\Psi_{t+1}^{\mathrm{UL}})^{-1}+(\Psi_{t+1}^{\mathrm{UL}})^{-1}\Psi_{t+1}^{\mathrm{UR}}Q_{t+1}\Psi_{t+1}^{\mathrm{LL}}(\Psi_{t+1}^{\mathrm{UL}})^{-1} & -(\Psi_{t+1}^{\mathrm{UL}})^{-1}\Psi_{t+1}^{\mathrm{UR}}Q_{t+1}\\ -Q_{t+1}\Psi_{t+1}^{\mathrm{LL}}(\Psi_{t+1}^{\mathrm{UL}})^{-1} & Q_{t+1}\end{bmatrix}, \tag{23}$$

where $Q_{t+1}=(\Psi_{t+1}^{\mathrm{LR}}-\Psi_{t+1}^{\mathrm{LL}}(\Psi_{t+1}^{\mathrm{UL}})^{-1}\Psi_{t+1}^{\mathrm{UR}})^{-1}$. Note that in order to give a uniform expression for Case 1 and Case 2, we introduce a matrix Ξ, which equals to a $(2l_t+u_t+1)$-dimensional identity matrix for Case 1 and an elementary matrix E_t for Case 2.

From (23), we notice that the sub-blocks Ψ_{t+1}^{LR}, Ψ_{t+1}^{LL} and Ψ_{t+1}^{UR} introduced by the new incoming data are relatively easy to compute except the matrix inverse $(\Psi_{t+1}^{\mathrm{UL}})^{-1}$. Comparing (19) and (21) with (13), we find that the UL sub-block of Ψ_{t+1}, Ψ_{t+1}^{UL} differs from Ψ_t, as P_t is replaced by $P_t+\triangle P_t$. Consequently, $(\Psi_{t+1}^{\mathrm{UL}})^{-1}$ is different from Ψ_t^{-1}. So, Ψ_{t+1}^{-1} cannot be directly updated from Ψ_t^{-1}. That is to say, the above mentioned method to obtain an incremental solution by decomposing Ψ_{t+1} is not achieved. Considering this, we would like to make a further decomposition of the sub-block Ψ_{t+1}^{UL}. To simplify the notation, we let $\bar{\Psi}_{t+1}$ denote Ψ_{t+1}^{UL}, and it can be decomposed into

$$\begin{aligned}\bar{\Psi}_{t+1}&=\left[\begin{array}{cc:c}0 & \boldsymbol{y}_t & \mathbf{0}_{1\times(l_t+u_t)}\\ \boldsymbol{y}_t & \frac{1}{C}I_{l_t\times l_t} & D(\boldsymbol{y}_t)J_tK_t\\ \hdashline \mathbf{0}_{(l_t+u_t)\times 1} & -K_tJ_t^TD(\boldsymbol{y}_t) & P_t+\triangle P_t\end{array}\right]\\ &\triangleq\left[\begin{array}{c:c}\bar{\Psi}_{t+1}^{\mathrm{UL}} & \bar{\Psi}_{t+1}^{\mathrm{UR}}\\ \hdashline \bar{\Psi}_{t+1}^{\mathrm{LL}} & \bar{\Psi}_{t+1}^{\mathrm{LR}}\end{array}\right],\end{aligned} \tag{24}$$

where

$$\begin{aligned}&\bar{\Psi}_{t+1}^{\mathrm{UL}}=\begin{bmatrix}0 & \boldsymbol{y}_t\\ \boldsymbol{y}_t & \frac{1}{C}I_{l_t\times l_t}\end{bmatrix},\quad \bar{\Psi}_{t+1}^{\mathrm{UR}}=\begin{bmatrix}\mathbf{0}_{1\times(l_t+u_t)}\\ D(\boldsymbol{y}_t)J_tK_t\end{bmatrix},\\ &\bar{\Psi}_{t+1}^{\mathrm{LL}}=\begin{bmatrix}\mathbf{0}_{(l_t+u_t)\times 1} & -K_tJ_t^TD(\boldsymbol{y}_t)\end{bmatrix},\quad \bar{\Psi}_{t+1}^{\mathrm{LR}}=P_t+\triangle P_t.\end{aligned} \tag{25}$$

Based on the block matrix inversion lemma, $\bar{\Psi}_{t+1}^{-1}$ can be computed by

$$\bar{\Psi}_{t+1}^{-1}=\begin{bmatrix}(\bar{\Psi}_{t+1}^{\mathrm{UL}})^{-1}+(\bar{\Psi}_{t+1}^{\mathrm{UL}})^{-1}\bar{\Psi}_{t+1}^{\mathrm{UR}}\bar{A}_{t+1}\bar{\Psi}_{t+1}^{\mathrm{LL}}(\bar{\Psi}_t^{\mathrm{UL}})^{-1} & -(\bar{\Psi}_{t+1}^{\mathrm{UL}})^{-1}\bar{\Psi}_{t+1}^{\mathrm{UR}}\bar{A}_{t+1}\\ -\bar{A}_{t+1}\bar{\Psi}_{t+1}^{\mathrm{LL}}(\bar{\Psi}_{t+1}^{\mathrm{UL}})^{-1} & \bar{A}_{t+1}\end{bmatrix}, \tag{26}$$

where $(\bar{\Psi}_{t+1}^{\mathrm{UL}})^{-1}$ is given by

$$(\bar{\Psi}_{t+1}^{\mathrm{UL}})^{-1}=\begin{bmatrix}-\frac{1}{Cl_t} & \frac{1}{l_t}\boldsymbol{y}_t^T\\ \frac{1}{l_t}\boldsymbol{y}_t & CI_{l_t\times l_t}-\frac{C}{l_t}\boldsymbol{y}_t\boldsymbol{y}_t^T\end{bmatrix}, \tag{27}$$

and

$$\bar{A}_{t+1}=(P_t+\triangle P_t-\bar{\Psi}_{t+1}^{\mathrm{LL}}(\bar{\Psi}_{t+1}^{\mathrm{UL}})^{-1}\bar{\Psi}_{t+1}^{\mathrm{UR}})^{-1}. \tag{28}$$

From (26), we can see that the terms $(\bar{\Psi}_{t+1}^{\mathrm{UL}})^{-1}$, $\bar{\Psi}_{t+1}^{\mathrm{UR}}$ and $\bar{\Psi}_{t+1}^{\mathrm{LL}}$ remain the same as their last values without additional computations. The only unknown part is $\bar{A}_{t+1}$, as

it relates to the new incoming data at time $t+1$. However, solving $\bar{A}_{t+1}$ needs the computation of $(l_t+u_t)\times(l_t+u_t)$ matrix inverse, which is still costly. According to the Sherman-Morrison-Woodbury formula [7], if $\triangle P_t$ can be decomposed into the product of low-dimensional matrices, $\bar{A}_{t+1}$ can be decoupled into two parts. One part is $A_t = (P_t - \bar{\Psi}^{\text{LL}}_{t+1}(\bar{\Psi}^{\text{UL}}_{t+1})^{-1}\bar{\Psi}^{\text{UR}}_{t+1})^{-1}$, which has already been computed and stored at time t. The other incremental part $\triangle A_t$ depending on the new added data must be easily computed. If it is possible, $\bar{A}_{t+1}$ can be efficiently updated from the last A_t without bringing in much computation. Unfortunately, from (18), we notice that $\triangle P_t$ contains the term of $K_t D(\boldsymbol{k}_{t+1}) K_t$, which is the product of three (l_t+u_t)-order full rank matrices. So, it cannot be decomposed into the product of low-dimensional matrices, and thus an efficient incremental S^{2}LS-SVM cannot be obtained if the standard manifold regularization is used.

The analysis in this section shows that the batch S^{2}LS-SVM proposed in [27] cannot be directly extended to an online implementation, if the standard manifold regularization is used.

4. Online S^{3}VM Algorithm

In this section, a new form of manifold regularization is proposed and then an online semi-supervised SVM algorithm, OS^{3}VM is developed.

4.1. Manifold Regularization using Fixed Kernel

As mentioned in Sec. 3, the main difficulty in the online implementation of S^{2}LS-SVM is that the term $\triangle P_t$ cannot be decomposed into the product of low-dimensional matrices, since the UL sub-block of the graph Laplacian matrix L_{t+1} in (16) changes from L_t once a new data is added to the current training set, if the standard manifold regularization is used. To solve this problem, in this paper, a new form of manifold regularization is proposed such that $\triangle P_t$ becomes decomposable. To be specific, we fix some anchor points in prior and regularize the manifold on these anchor points instead of the whole training data set. It is obvious that if the anchor points are good representatives of the whole data set, the manifold regularization with respect to the anchor points can give a good approximation of the manifold of the whole data set to some extent.

Now, the problem comes how to choose the anchor points? Based on the smoothness assumption, the samples that are close in the high-density region should also have similar outputs (labels). So, if the selected anchor points can generally cover the high-density region, then the smoothness of the outputs (the manifold) can be ensured to some extent. To achieve this, a typical strategy is to select the anchor points uniformly, and then fix them in the latter iterations. If the number of anchor points is suitable, the high-density region can be roughly covered. Another more efficient way is to use the method of vector quantization proposed in [5, 6], that is, select more anchor points (centers of data) from the high-density region but less from the low-density region according to the distribution of data. This method is more reasonable, but costly to be implemented online due to the complex computation in the updation of anchor points at each time. Considering this, we use the former one in this paper.

Once the anchor points are selected, according to the smoothness assumption, the predicted output of each data sample can be approximated by the weighted average of the outputs of N anchor points [20], which is given by

$$\hat{f}(\boldsymbol{x}_i) = \sum_{j=1}^{N} W_{ij} f(\boldsymbol{c}_j), \tag{29}$$

where $\hat{f}(\boldsymbol{x}_i)$ denotes the predicted output of $\boldsymbol{x}_i$ based on the anchor points. The notations $\boldsymbol{c}_j$ and $f(\boldsymbol{c}_j)$ denote the j-th anchor point and its corresponding predicted output, respectively. The coefficient W_{ij} denotes the normalized weight that measures the similarity between $\phi(\boldsymbol{x}_i)$ and $\phi(\boldsymbol{c}_j)$, and satisfies $\sum_{j=1}^{N} W_{ij} = 1$.

Based on (29), we propose a new form of manifold regularization to constrain the smoothness with respect to the anchor points, which is expressed as

$$\begin{aligned} \|f\|_{\mathcal{M}}^2 &= \sum_{i=1}^{l+u} (f(\boldsymbol{x}_i) - \sum_{j=1}^{N} W_{ij} f(\boldsymbol{c}_j))^2 \\ &= \boldsymbol{\alpha}^T (KK - KW\bar{K}^T - \bar{K}W^T K + \bar{K}W^T W\bar{K}^T)\boldsymbol{\alpha}, \end{aligned} \tag{30}$$

where K is the same as that define in (3), W denotes a $(l+u)\times N$ weight matrix with its ij-th element W_{ij}, and $\bar{K}$ is a $(l+u)\times N$ matrix with its ij-th element $\bar{k}_{ij} =< \phi(\boldsymbol{x}_i), \phi(\boldsymbol{c}_j) >$ represents the inner product of an input vector $\boldsymbol{x}_i$ and an anchor point $\boldsymbol{c}_j$ mapping to the RKHS. It is noted that by fixing a few number of anchor points $\boldsymbol{c}_j$, the center of the kernel function is fixed. In the following, we name the above regularization term $\|f\|_{\mathcal{M}}^2$ as the *manifold regularization using fixed kernel*.

Since the number of anchor points is usually much smaller than the total number of data samples, i.e. $N << l+u$, the standard manifold regularization is simplified. Besides, it is shown in the following, the increment $\triangle P_t$ can be easily divided into the product of low-dimensional matrices such that an efficient online S^{2}LS-SVM algorithm can be obtained, if the new manifold regularization is used.

4.2. OS^{3}VM with Manifold Regularization Using Fixed Kernel

In this subsection, based on the proposed manifold regularization given in Sec. 4.1, the information of new incoming data (labeled or unlabeled) is well exploited, and a new LS-based online S^{3}VM (OS^{3}VM) algorithm is developed. This algorithm mainly includes two steps, *incremental step* and *decremental step*. The former is used to update the solution with new incoming data. The latter is used to omit the considered irrelevant data so as to control the computational complexity and the space for storing the current training data set. Next, we present these two steps in detail respectively.

4.2.1. Incremental Step

According to the framework of online S^{2}LS-SVM given in Sec. 3 at each time instant t, the instantaneous LAEs satisfying the KKT conditions of the proposed OS^{3}VM can be obtained, which are similar to (11). But, as the new manifold regularization (30)

is used in the OS^{3}VM, the instantaneous P_t in parameter matrix Ψ_t is changed, which is given by

$$P_t = \gamma_{\mathcal{H}} K_t + \gamma_{\mathcal{M}}(K_t K_t - K_t W_t \bar{K}_t^T - \bar{K}_t W_t^T K_t + \bar{K}_t W_t^T W_t \bar{K}_t^T). \tag{31}$$

At the next time $t+1$, the new incoming data is added to the current training data set, and the incremental relationships between $K_{t+1}, \bar{K}_{t+1}, W_{t+1}$ and their corresponding last values become

$$K_{t+1} = \begin{bmatrix} K_t & \boldsymbol{k}_{t+1} \\ \boldsymbol{k}_{t+1}^T & 1 \end{bmatrix}, \quad \bar{K}_{t+1} = \begin{bmatrix} \bar{K}_t \\ \bar{\boldsymbol{k}}_{t+1} \end{bmatrix}, \quad W_{t+1} = \begin{bmatrix} W_t \\ \boldsymbol{w}_{t+1} \end{bmatrix}, \tag{32}$$

where K_{t+1} is the same as that given in Sec. 3, $\bar{\boldsymbol{k}}_{t+1} = [k(\boldsymbol{x}_{t+1}, \boldsymbol{c}_1), \ldots, k(\boldsymbol{x}_{t+1}, \boldsymbol{c}_N)]$, and $\boldsymbol{w}_{t+1} = [W_{t+1,1}, \ldots, W_{t+1,N}]$.

Based on (32), we have

$$\begin{aligned} P_{t+1} &= \gamma_{\mathcal{H}} K_{t+1} + \gamma_{\mathcal{M}}(K_{t+1} K_{t+1} - K_{t+1} W_{t+1} \bar{K}_{t+1}^T - \bar{K}_{t+1} W_{t+1}^T K_{t+1} + \bar{K}_{t+1} W_{t+1}^T W_{t+1} \bar{K}_{t+1}^T) \\ &\triangleq \begin{bmatrix} P_{t+1}^{\mathrm{UL}} & P_{t+1}^{\mathrm{UR}} \\ P_{t+1}^{\mathrm{LL}} & P_{t+1}^{\mathrm{LR}} \end{bmatrix}, \end{aligned} \tag{33}$$

where the four sub-blocks of P_{t+1} are given by

$$\begin{aligned} P_{t+1}^{\mathrm{UL}} &= [\gamma_{\mathcal{H}} K_t + \gamma_{\mathcal{M}}(K_t K_t - K_t W_t \bar{K}_t^T - \bar{K}_t W_t^T K_t + \bar{K}_t W_t^T W_t \bar{K}_t^T)] \\ &\quad + \gamma_{\mathcal{M}}(\boldsymbol{k}_{t+1} \boldsymbol{k}_{t+1}^T - \boldsymbol{k}_{t+1} \boldsymbol{w}_{t+1} \bar{K}_t^T - \bar{K}_t \boldsymbol{w}_{t+1}^T \boldsymbol{k}_{t+1}^T + \bar{K}_t \boldsymbol{w}_{t+1}^T \boldsymbol{w}_{t+1} \bar{K}_t^T), \\ &\triangleq P_t + \triangle P_t, \\ P_{t+1}^{\mathrm{UR}} &= \gamma_{\mathcal{H}} \boldsymbol{k}_{t+1} + \gamma_{\mathcal{M}}(K_t \boldsymbol{k}_{t+1} + \boldsymbol{k}_{t+1} - K_t W_t \bar{\boldsymbol{k}}_{t+1}^T - \bar{K}_t W_t^T \boldsymbol{k}_{t+1} \\ &\quad - \bar{K}_t \boldsymbol{w}_{t+1}^T - \boldsymbol{k}_{t+1} \boldsymbol{w}_{t+1} \bar{\boldsymbol{k}}_{t+1}^T + \bar{K}_t W_t^T W_t \bar{\boldsymbol{k}}_{t+1}^T + \bar{K}_t \boldsymbol{w}_{t+1}^T \boldsymbol{w}_{t+1} \bar{\boldsymbol{k}}_{t+1}^T), \\ P_{t+1}^{\mathrm{LL}} &= \gamma_{\mathcal{H}} \boldsymbol{k}_{t+1}^T + \gamma_{\mathcal{M}}(\boldsymbol{k}_{t+1}^T K_t + \boldsymbol{k}_{t+1}^T - \boldsymbol{k}_{t+1}^T W_t \bar{K}_t^T - \bar{\boldsymbol{k}}_{t+1} W_t^T K_t - \boldsymbol{w}_{t+1} \bar{K}_t^T \\ &\quad - \bar{\boldsymbol{k}}_{t+1} \boldsymbol{w}_{t+1}^T \boldsymbol{k}_{t+1}^T + \bar{\boldsymbol{k}}_{t+1} W_t^T W_t \bar{K}_t^T + \bar{\boldsymbol{k}}_{t+1} \boldsymbol{w}_{t+1}^T \boldsymbol{w}_{t+1} \bar{K}_t^T), \\ P_{t+1}^{\mathrm{LR}} &= \gamma_{\mathcal{H}} + \gamma_{\mathcal{M}}(1 + \boldsymbol{k}_{t+1}^T \boldsymbol{k}_{t+1} - \boldsymbol{k}_{t+1}^T W_t \bar{\boldsymbol{k}}_{t+1}^T - \bar{\boldsymbol{k}}_{t+1} W_t^T \boldsymbol{k}_{t+1} - \bar{\boldsymbol{k}}_{t+1} \boldsymbol{w}_{t+1}^T \\ &\quad - \boldsymbol{w}_{t+1} \bar{\boldsymbol{k}}_{t+1}^T + \bar{\boldsymbol{k}}_{t+1} W_t^T W_t \bar{\boldsymbol{k}}_{t+1}^T + \bar{\boldsymbol{k}}_{t+1} \boldsymbol{w}_{t+1}^T \boldsymbol{w}_{t+1} \bar{\boldsymbol{k}}_{t+1}^T), \end{aligned}$$

with $\triangle P_t$ in P_{t+1}^{UL} being

$$\triangle P_t = \gamma_{\mathcal{M}}(\boldsymbol{k}_{t+1} \boldsymbol{k}_{t+1}^T - \boldsymbol{k}_{t+1} \boldsymbol{w}_{t+1} \bar{K}_t^T - \bar{K}_t \boldsymbol{w}_{t+1}^T \boldsymbol{k}_{t+1}^T + \bar{K}_t \boldsymbol{w}_{t+1}^T \boldsymbol{w}_{t+1} \bar{K}_t^T). \tag{34}$$

From (34), we can see that the rank of $\triangle P_t$ reduces to 1. So, it can be decomposed into the product of two vectors, $P_t = V_t U_t^T$, where $V_t = \gamma_{\mathcal{M}}(\boldsymbol{k}_{t+1} - \bar{K}_t \boldsymbol{w}_{t+1}^T) \in \mathbb{R}^{(l_t+u_t)\times 1}$ and $U_t^T = (\boldsymbol{k}_{t+1}^T - \boldsymbol{w}_{t+1} \bar{K}_t^T) \in \mathbb{R}^{1\times(l_t+u_t)}$. Based on this decomposition, an incremental solution of $\bar{A}_{t+1}$ can be obtained as follow

$$\begin{aligned} \bar{A}_{t+1} &= \left((P_t - \bar{\Psi}_{t+1}^{\mathrm{LL}} (\bar{\Psi}_{t+1}^{\mathrm{UL}})^{-1} \bar{\Psi}_{t+1}^{\mathrm{UR}}) + \triangle P_t\right)^{-1} \\ &= \left((P_t - \bar{\Psi}_{t+1}^{\mathrm{LL}} (\bar{\Psi}_{t+1}^{\mathrm{UL}})^{-1} \bar{\Psi}_{t+1}^{\mathrm{UR}}) + V_t U_t^T\right)^{-1} \\ &= A_t - A_t V_t (1 + U_t^T A_t V_t)^{-1} U_t^T A_t, \end{aligned}$$

where $A_t = (P_t - \bar{\Psi}^{\text{LL}}_{t+1}(\bar{\Psi}^{\text{UL}}_{t+1})^{-1}\bar{\Psi}^{\text{UR}}_{t+1})^{-1}$ is available at time $t+1$ without additional computation, while the incremental part

$$\triangle A_t = -A_t V_t (1 + U_t^T A_t V_t)^{-1} U_t^T A_t, \tag{35}$$

should be computed. It is remarked that since the term $(1 + U_t^T A_t V_t)$ is a scalar, its inverse is easy to compute.

Substituting $\bar{A}_{t+1} = A_t + \triangle A_t$ into (26), an incremental solution of $\bar{\Psi}^{-1}_{t+1}$ can be obtained as

$$\bar{\Psi}^{-1}_{t+1} = \Psi^{-1}_t + \begin{bmatrix} (\bar{\Psi}^{\text{UL}}_{t+1})^{-1}\bar{\Psi}^{\text{UR}}_{t+1}\triangle A_t \bar{\Psi}^{\text{LL}}_{t+1}(\bar{\Psi}^{\text{UL}}_{t+1})^{-1} & -(\bar{\Psi}^{\text{UL}}_{t+1})^{-1}\bar{\Psi}^{\text{UR}}_{t+1}\triangle A_t \\ -\triangle A_t \bar{\Psi}^{\text{LL}}_{t+1}(\bar{\Psi}^{\text{UL}}_{t+1})^{-1} & \triangle A_t \end{bmatrix}. \tag{36}$$

Once $\bar{\Psi}^{-1}_{t+1}$ is obtained, Ψ^{-1}_{t+1} can be easily computed. Similar to Sec. 3, according to types of data (labeled or unlabeled), there are two different cases.

Case 1: The incoming sample is unlabeled.

Base on $\bar{\Psi}^{-1}_{t+1}$ given in (36) and the block matrix inversion lemma [9, 10], Ψ^{-1}_{t+1} can be obtained as

$$\Psi^{-1}_{t+1} = \begin{bmatrix} \bar{\Psi}^{-1}_{t+1} + \bar{\Psi}^{-1}_{t+1}\Psi^{\text{UR}}_{t+1} Q_{t+1} \Psi^{\text{LL}}_{t+1}\bar{\Psi}^{-1}_{t+1} & -\bar{\Psi}^{-1}_{t+1}\Psi^{\text{UR}}_{t+1} Q_{t+1} \\ -Q_{t+1}\Psi^{\text{LL}}_{t+1}\bar{\Psi}^{-1}_{t+1} & Q_{t+1} \end{bmatrix}, \tag{37}$$

where $Q_{t+1} = (\Psi^{\text{LR}}_{t+1} - \Psi^{\text{LL}}_{t+1}\bar{\Psi}^{-1}_{t+1}\Psi^{\text{UR}}_{t+1})^{-1}$ is a scalar, thus easy to be computed.

Case 2: The incoming sample is labeled.

Similar to Case 1, the reordered Ψ_{t+1}, i.e. $E_t\Psi_{t+1}E_t$ that incorporates new P_{t+1} can be decomposed into four parts given in (21). Consequently, the matrix inverse of Ψ_{t+1} becomes

$$\Psi^{-1}_{t+1} = E_t \begin{bmatrix} \bar{\Psi}^{-1}_{t+1} + \bar{\Psi}^{-1}_{t+1}\Psi^{\text{UR}}_{t+1} Q_{t+1} \Psi^{\text{LL}}_{t+1}\bar{\Psi}^{-1}_{t+1} & -\bar{\Psi}^{-1}_{t+1}\Psi^{\text{UR}}_{t+1} Q_{t+1} \\ -Q_{t+1}\Psi^{\text{LL}}_{t+1}\bar{\Psi}^{-1}_{t+1} & Q_{t+1} \end{bmatrix} E_t, \tag{38}$$

where $Q_{t+1} = (\Psi^{\text{LR}}_{t+1} - \Psi^{\text{LL}}_{t+1}\bar{\Psi}^{-1}_{t+1}\Psi^{\text{UR}}_{t+1})^{-1}$ is a 2×2 matrix, as the parameter matrix increases by two dimensions for this case. It is also remarked that the multiplication of the elementary matrix E_t does not need additional computation, as it can be easily implemented by exchanging the corresponding rows and columns.

Remark 2: As mentioned in **Remark 1**, in order to simplify the computation, the S^{2}LS-SVM proposed in [27] ignored the term $\triangle P_t$. Since $\triangle P_t$ in (18) contains the useful information brought by the new incoming data, such a simplification ignores a lot of useful information underlying the new incoming data. Consequently, the performance of classification, especially when the labeled data are scarce, is further deteriorated. In this paper, not only the impact of new arrival data on P_{t+1} is well considered, but also the information of new incoming unlabeled data is fully exploited. So, a more accurate incremental solution can be obtained. Besides, as discussed in Case 1 and Case 2, the OS^{3}VM can avoid the complex computation of high-dimensional matrix inversion.

4.2.2. Decremental Step

Although the incremental step presented above can ensure good classification performance, the burden of computation also increases with the increasing number of data. So, it is inefficient to deal with the large-scale data classification problems.

Considering this, similar to the online LS-SVM [11] and the S^{2}LS-SVM [27], we also perform a decremental step, which is used to omit those considered less irrelevant samples and fix the number of currently used training data as N_{thre} in each iteration. Generally speaking, the training data corresponding to the element with the smallest absolute value in $\boldsymbol{\alpha}_t$ is considered as the irrelevant sample, as it has little effect on the discriminant function f [11, 27]. So, it is removed from the training set in the decremental step.

It is noted that, as the reverse process of the incremental step, the decremental step also has two main steps, that is, one step is used to omit the considered irrelevant sample, and the other step is used to remove its influence on $\triangle P_t$ from P_{t+1}. As to the first step, according to the types of the pruned data samples, we have the following two cases.

Case 1: The pruned sample is unlabeled.

We let k denote the index corresponding to the element with the smallest absolute value in $\boldsymbol{\alpha}_t$, which means that the k-th data sample must be removed from the training set. We assume that the q-th element of λ_t corresponds to the k-th element of $\boldsymbol{\alpha}_t$. Reversing to (37) in the incremental step, if the k-th data is removed, the matrix inverse $\bar{\Psi}_{t+1}^{-1}$ must be updated by

$$\bar{\Psi}_{t+1}^{-1} = \Psi_{t,R_uR_u}^{-1} - \Psi_{t,R_uM_u}^{-1}(\Psi_{t,M_uM_u}^{-1})^{-1}\Psi_{t,M_uR_u}^{-1}, \tag{39}$$

where $\Psi_{t,R_uM_u}^{-1}$ denotes the submatrix of Ψ_t^{-1} that consists of those rows given in set $R_u = \{1, \ldots, q-1, q+1, \ldots, 1+2l_t+u_t\}$ and those columns given in set $M_u = \{q\}$. Since $\Psi_{t,M_uM_u}^{-1}$ is a scalar, its inverse is easy to compute.

Case 2: The pruned sample is labeled.

In this case, since the pruned sample is labeled, we should remove the corresponding elements in both $\boldsymbol{\alpha}_t$ and $\boldsymbol{\beta}_t$. In this case, $\bar{\Psi}_{t+1}^{-1}$ is reduced to a $(2l_t+u_t-1)\times(2l_t+u_t-1)$ matrix. Let q_1 and q_2 denote the q_1-th element of λ_t (i.e. (q_1-1)-th element of $\boldsymbol{\beta}_t$) and the q_2-th element of λ_t (i.e. k-th element of $\boldsymbol{\alpha}_t$), respectively. Reversing to (38) in the incremental step, we have

$$\bar{\Psi}_{t+1}^{-1} = \Psi_{t,R_lR_l}^{-1} - \Psi_{t,R_lM_l}^{-1}(\Psi_{t,M_lM_l}^{-1})^{-1}\Psi_{t,M_lR_l}^{-1}, \tag{40}$$

where $\Psi_{t,R_lM_l}^{-1}$ denotes the submatrix of Ψ_t^{-1} that consists of those rows given in set $R_l = \{1, \ldots, q_1-1, q_1+1, \ldots, q_2-1, q_2+1, \ldots, 1+2l_t+u_t\}$ and those columns given in set $M_l = \{q_1, q_2\}$. In (40), the only unknown is the term $(\Psi_{t,M_lM_l}^{-1})^{-1}$ at the current time $t+1$. Because $\Psi_{t,M_lM_l}^{-1}$ is 2×2 matrix, it does not take much computation to obtain its inverse.

For the second step, we should remove the effect of $\triangle P_t$ from P_{t+1} when a sample is pruned. Reversing to (36) in the incremental step for both two types of data, we have

$$\Psi_{t+1}^{-1} = \bar{\Psi}_{t+1}^{-1} - \begin{bmatrix} (\bar{\Psi}_{t+1}^{\text{UL}})^{-1}\bar{\Psi}_{t+1}^{\text{UR}}\triangle A_t\bar{\Psi}_{t+1}^{\text{LL}}(\bar{\Psi}_{t+1}^{\text{UR}})^{-1} & -(\bar{\Psi}_{t+1}^{\text{UL}})^{-1}\bar{\Psi}_{t+1}^{\text{UR}}\triangle A_t \\ -\triangle A_t\bar{\Psi}_{t+1}^{\text{LL}}(\bar{\Psi}_{t+1}^{\text{UL}})^{-1} & \triangle A_t \end{bmatrix}, \tag{41}$$

where $\bar{\Psi}_{t+1}^{-1}$ is given in (39) for Case 1 or (40) for Case 2.

By performing the incremental step and the decremental step alternatively, the OS^{3}VM algorithm can adapt to sequential data classification without consuming much

computation. It can not only incorporate the information of new incoming data efficiently, but also delete the considered irrelevant data sample. The detailed implementation procedure of OS^{3}VM is summarized in Algorithm 1.

Algorithm 1 OS^{3}VM Algorithm

Initialization: Initialize parameter matrices Ψ_0, Ψ_0^{-1}, Y_0, and the current number of training data n_0.

for $t = 0, 1, 2, \ldots$ **do**
  if $n_t < N_{\text{thre}}$, ▷ **incremental step**
    if the incoming data is **unlabeled**, **then**
      $\{\boldsymbol{x}_{t+1}\}$ is **added** to the training data set.
      Update $\bar{\Psi}_{t+1}^{-1}$ via (36).
      Update Ψ_{t+1}^{-1} via (37).
    end if
    if the incoming data is **labeled**, **then**
      $\{\boldsymbol{x}_{t+1}, y_{t+1}\}$ is **added** to the training data set.
      Update $\bar{\Psi}_{t+1}^{-1}$ via (36).
      Update Ψ_{t+1}^{-1} via (38).
    end if
    $n_{t+1} = n_t + 1$.
  else $n_t \geq N_{\text{thre}}$, ▷ **decremental step**
    if the pruned data is **unlabeled**, **then**
      $\{\boldsymbol{x}_k\}$ is **removed** from the training data set.
      Update $\bar{\Psi}_{t+1}^{-1}$ via (39).
      Update Ψ_{t+1}^{-1} via (41).
    end if
    if the pruned data is **labeled**, **then**
      $\{\boldsymbol{x}_k, y_k\}$ is **pruned** from the training data set.
      Update $\bar{\Psi}_{t+1}^{-1}$ via (40).
      Update Ψ_{t+1}^{-1} via (41).
    end if
    $n_{t+1} = n_t - 1$.
  end if
  Compute $\boldsymbol{\beta}_{t+1}$, $\boldsymbol{\alpha}_{t+1}$ and b_{t+1};
end for

5. Experimental Results

In this section, a series of experiments are performed to show the effectiveness of the proposed OS^{3}VM algorithm. For the purpose of comparison, the result of LS-SVM algorithm proposed in [11], and S^{2}LS-SVM algorithm proposed in [27] are also given. For each experiment, we perform the independent Monte Carlo simulations for 100 times, and the averaged results over 100 simulations are given in the following. To measure the performance of different algorithms, the misclassification rate (MCR) [13]

is used, which is defined as

$$\text{MCR} = \frac{\text{the number of mis-classified samples}}{\text{the total number of samples}} \times 100\%. \tag{42}$$

All the experiments are performed using MATLAB 2014a on the same machine with four-core (3.4-GHz) CPU, 16.0-GB RAM, and Windows 10 operation system.

5.1. Classification for Synthetic Data Set

Firstly, the classification of "Double-Moon" data set is considered. In this problem, there are two attributes for each data sample, which corresponds to the two coordinates in the horizontal and vertical axises respectively. The aim is to find a discriminant function such that the data points can be classified into two classes, *the upper moon* and *the lower moon*.

At each trail of simulation, we select 10 samples as anchor points, total 240 samples containing 16 labeled and 224 unlabeled samples for training, 20 samples for testing. The zero-mean Gaussian white noise is also added to the samples such that the signal-to-noise ratios (SNR) equals to 20 dB. In the initial stage, the training data set contains two labeled samples and six unlabeled samples. Then, at each time, one data randomly selected from the training data set is added to the current training data set. In the simulation, the Gaussian kernel function is used with kernel width $\sigma = 5$. Besides, the parameters are set as $C = 50$, $\gamma_{\mathcal{H}} = 0.01$, $\gamma_{\mathcal{M}} = 0.05$ and $N_{\text{thre}} = 20$.

The learning curves of the considered three algorithms are shown in Fig. 1. We can see that the MCRs of these three algorithms decrease drastically initially. After about 300 iterations, they converge and keep nearly invariant. In the initial a few iterations, since both OS^{3}VM and S^{2}LS-SVM consider the information of unlabeled data, their performances are similar and both are much better than that of the LS-SVM. However, due to the simplification of computation and lack of the information of new incoming unlabeled data in the latter iterations, S^{2}LS-SVM becomes inferior to the OS^{3}VM after about 20 iterations. From Fig. 1, we find that the final MCR of the OS^{3}VM is smaller than that of the other two algorithms by about 5%.

Besides, in order to show the efficiency of the proposed algorithm, we have compared the computational complexity in each iteration (including both the incremental step and the decremental step) of OS^{3}VM with the other two algorithms. Assume the numbers of labeled data and unlabeled data in the training dataset at the time instant t are l_t are u_t respectively. For OS^{3}VM algorithm, computing the matrix inverse Ψ_{t+1}^{-1} requires $O(4(2l_t + u_t + 1)^3)$ multiplication operations and $O(4(2l_t + u_t + 1)^3)$ addition operations in each iteration respectively. For the S^{2}LS-SVM, updating the matrix Ψ_{t+1}^{-1} requires $O(2(2l_t + u_t + 1)^3$ multiplication operations and $O(2(2l_t + u_t + 1)^3)$ addition operations. As to the LS-SVM, since the unlabeled data are ignored, we only need $O(2(2l_t + 1)^3)$ multiplication operations and $O(2(2l_t + 1)^3)$ addition operations to compute Ψ_{t+1}^{-1} in each iteration. From the above analyses, we find that the computational complexity of our proposed OS^{3}VM is about twice that of the other two algorithms, as the effects of the new incoming labeled data and unlabeled data are fully considered. Recalling that the size of the current training data sets $l_t + u_t$ is restricted to a certain threshold N_{thre}, the total number of computations in each iteration of OS^{3}VM cannot

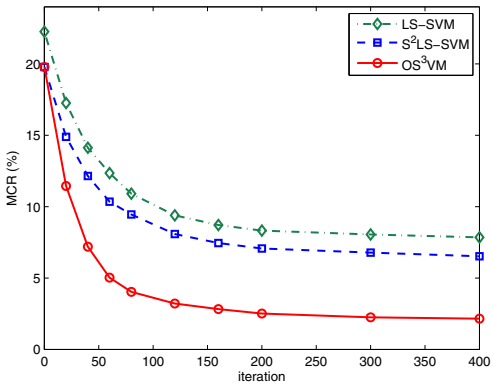

Figure 1: The learning curves of different algorithms for "Double-Moon" classification problem.

be significantly larger than that of the other two algorithms, if a suitable small N_{thre} is selected.

On the other hand, from Fig. 1, we also can see that the MCR of the OS^{3}VM becomes smaller than 10% after about 30 iterations, while S^{2}LS-SVM and LS-SVM respectively need about 70 and 100 iterations to achieve the same performance level. That is to say, although OS^{3}VM needs more computations in each iteration, it converges fastest owing to the full consideration of the new incoming data. So, for fair comparison, we record the running times of different algorithms when their MCRs achieve the same level (below to 10% in the simulation). The running times are 0.232, 0.209, and 0.295 seconds for the OS^{3}VM, the S^{2}LS-SVM and the LS-SVM respectively. The results suggest that although OS^{3}VM takes more computations in each iteration, owing to its fast convergence rate, its running time for achieving the same classification performance is only slightly longer than that of the S^{2}LS-SVM, while shorter than that of the LS-SVM.

Fig. 2 shows the steady-state MCRs of different algorithms with different number of labeled samples, in which the steady-state MCRs are obtained by averaging the results over the last 100 samples after 300 iterations. Generally speaking, as the number of labeled data increases, the steady-state MCRs of these three algorithms become smaller and smaller, indicating the classification accuracy becomes higher and higher. When the training set contains more than 24 labeled data, the steady-state MCRs keep nearly invariant. From Fig. 2, we also find that the OS^{3}VM outperforms the other two algorithms in all the cases. When the labeled data are scarce, the performances of both OS^{3}VM and S^{2}LS-SVM are significantly better than that of the LS-SVM, which confirms that the unlabeled data can boost learning performance. When more than 20 labeled data are added to the training set, the steady-state MCR of the LS-SVM decreases rapidly and even becomes smaller than that of the S^{2}LS-SVM. The possible reason for this phenomenon is that the negative effect caused by the simplification of computation dominates the positive effect benefited from the incorporation of unlabeled data. In addition, as the number of labeled data increases, the performance

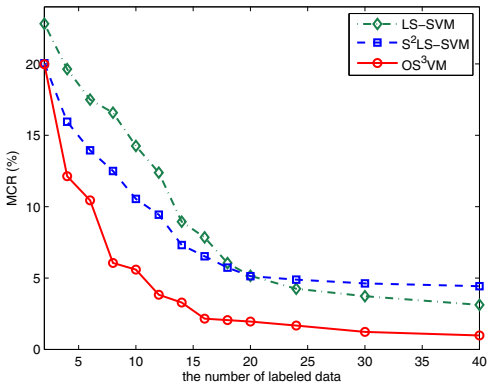

Figure 2: Steady-state MCRs versus the numbers of labeled data for "Double-Moon" classification problem.

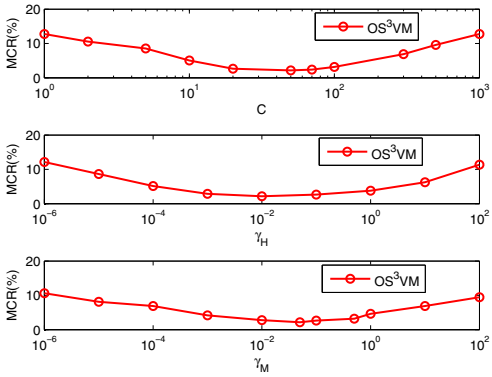

Figure 3: Steady-state MCRs versus different values of parameters C, $\gamma_{\mathcal{H}}$ and $\gamma_{\mathcal{M}}$ of OS^{3}VM for "Double-Moon" classification problem.

improvement by exploiting the information of unlabeled data declines, which makes the steady-state MCR of LS-SVM closes to that of OS^{3}VM.

We also test the effect of the parameters C, $\gamma_{\mathcal{H}}$ and $\gamma_{\mathcal{M}}$ on the performance of OS^{3}VM in Fig. 3. Similar changing trends are observed for different parameters. To be specific, initially, the MCR decreases as the parameter increases until a certain optimal value. After that, further increasing the value of the parameters, the MCR increases. Referring to the results given in Fig. 3, suitable choices of C, $\gamma_{\mathcal{H}}$ and $\gamma_{\mathcal{M}}$ should lie within $(10^1, 10^2)$, $(10^{-2}, 10^{-1})$, $(10^{-2}, 10^{-1})$, respectively.

The impact of N_{thre} on the classification performance of different algorithms is also investigated, and the results are given in Fig. 4. We can see that the OS^{3}VM has the best performance among these three algorithms in all the cases. Besides, we find

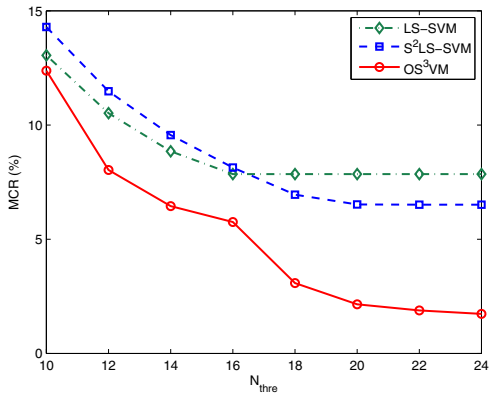

Figure 4: Steady-state MCRs versus N_{thre} for "Double-Moon" classification problem.

that the performance of OS^{3}VM depends significantly on N_{thre}, but that of the LS-SVM and S^{2}LS-SVM do not. For the supervised LS-SVM algorithm, it only employs the information of 16 labeled data. So, if N_{thre} is more than 16, the MCR of LS-SVM does not change much. For the S^{2}LS-SVM algorithm, since the information of new incoming unlabeled data is not considered, its MCR is less sensitive to the change of N_{thre}. In addition, from Fig. 4, we find that the performance of S^{2}LS-SVM is even worse than that of the LS-SVM when $N_{\text{thre}} < 18$ due to its (over) simplification in computation as explained in **Remark 1**. However, with the increasing of N_{thre}, the utilization of unlabeled data plays a positive role, making the S^{2}LS-SVM outperform the LS-SVM. The simulation results also provide us some guideline on how to select the parameter N_{thre}. From Fig. 4, the steady-state MCR of the OS^{3}VM decreases significantly when $N_{\text{thre}} < 20$. After that, the rate of decline becomes slower and slower. So, a suitable N_{thre} can be selected between 20 and 24.

5.2. *Classification for Real Data Sets*

In this subsection, some real data sets are adopted for testing the performance of the proposed algorithm. In the first simulation, the Columbia Object Image Library (COIL-20) is considered, which is a database of gray-scale images of 20 objects [19]. In this data set, each sample is a 32×32 image of one object taken from a specific angle. Note that as a two classification algorithm is considered in this paper, total 144 data samples of two classes are selected. For each data sample, there are 1024 attributes, which correspond to the gray scales of all the pixels. Our purpose is to classify the gray images into two classes.

At each trail of simulation, we select 10 data as anchor points, 110 data (12 labeled data and 98 unlabeled data) for training, and the remaining 24 data for testing. Each data is randomly added to the current training set whenever $n_t < N_{\text{thre}}$. In addition, the parameters remain the same as those used in the last example.

In Fig. 5, we depict the learning curves of the considered three algorithms. The results show that all the algorithms can converge after about 500 iterations and the

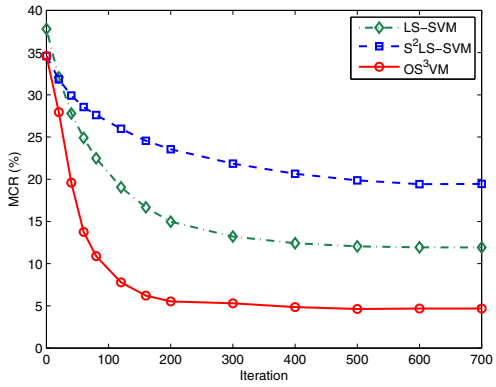

Figure 5: The learning curves of different algorithms for COIL20 classification problem.

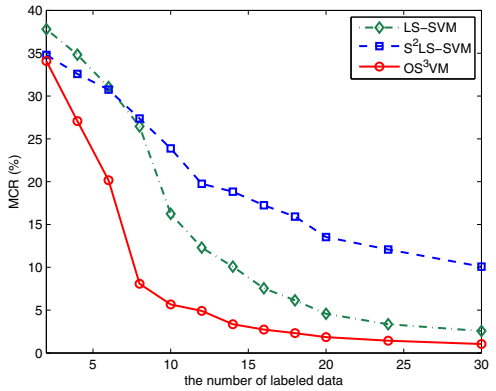

Figure 6: Steady-state MCRs versus the numbers of labeled data for COIL20 classification problem.

OS^3VM algorithm converges fastest. The steady-state MCR of OS^3VM is smaller than that of the LS-SVM algorithm by about 7%, and smaller than that of the S^2LS-SVM algorithm by about 15%. Different from the above example, S^2LS-SVM outperforms LS-SVM only in the initial a few iterations. The possible reason is illustrated as follows. The ratio between the number of labeled data and that of unlabeled data is relatively high in this example, which leads to the following two outcomes. On the one hand, considering that simplification in the computation is performed on labeled data, the negative effect of simplification becomes more and more significant as the number of labeled data increases. On the other hand, as the labeled data usually provide more useful information than the unlabeled data, the performance improvement gained from unlabeled data becomes weaker and weaker as the number of labeled data increases. It is also noted that, as the S^2LS-SVM and the LS-SVM perform rather poor (their MCRs

Table 1: The steady-state MCRs (%) versus the numbers of labeled data for Iris plant classification data

Algorithm	Number of labeled data					
	4	8	12	16	20	24
LS-SVM	11.75	8.31	5.62	2.50	1.05	0.62
S^2LS-SVM	9.88	7.50	6.34	5.75	5.36	5.27
OS^3VM	**9.73**	**5.25**	**3.13**	**1.88**	**0.62**	**0.45**

Table 2: The steady-state MCRs (%) versus the numbers of labeled data for air pollution classification data

Algorithm	Number of labeled data					
	4	8	12	16	20	24
LS-SVM	11.38	8.57	3.85	1.63	1.06	0.35
S^2LS-SVM	8.27	6.01	4.71	3.97	3.82	3.73
OS^3VM	**8.15**	**4.38**	**2.34**	**0.54**	**0.32**	**0.15**

Table 3: The steady-state MCRs (%) versus the numbers of labeled data for Banknote classification data

Algorithm	Number of labeled data					
	4	8	12	16	20	24
LS-SVM	14.56	10.19	6.75	3.68	2.50	1.87
S^2LS-SVM	12.43	9.10	7.37	6.39	6.05	5.86
OS^3VM	**10.08**	**6.04**	**4.32**	**2.85**	**2.08**	**1.43**

after convergence are always larger than 10%), the comparison for running times of different algorithms is not given here.

In Fig. 6, we also depict the steady-state MCRs of different algorithms with different numbers of labeled data. From Fig. 6, we can see that the changing trends are in general similar to Fig. 2 in the last example. The performances of all the algorithms are improved rapidly as the number of labeled data increases, and the OS^3VM achieves the best performance among these three algorithms in each case.

In order to further verify the effectiveness of the proposed algorithm, we also test the proposed algorithm on three other real datasets, the Iris plant dataset [1, 13], the atmosphere quality dataset [22], and the banknote dataset [1].

- The Iris plant is perhaps the best known dataset, which is frequently used in the context of pattern recognition [1, 13]. In this data set, there are total 150 samples consisting of 3 classes, and each class refers to a type of iris plant. For each data sample, there are 4 attributes, which correspond to the sepal length, the sepal width, the petal length and the petal width, respectively. Note that since the proposed algorithm is a two-classes classification algorithm, only two classes (or 100 samples) are employed in this experiment. At each trail of simulation, we randomly select 10 data samples as anchor points, 70 data for training, the remaining 20 data for testing.

- The atmosphere quality dataset is a binary classification data set, which is provided by [22]. This real dataset consists of 800 measurements of quality of air samples, 400 for clean air samples and 400 for slightly polluted air samples.

Table 4: The running time (sec) versus the numbers of labeled data for Iris plant classification data

Algorithm	Number of labeled data					
	4	8	12	16	20	24
LS-SVM	–	0.476	0.347	0.268	**0.182**	**0.138**
S^2LS-SVM	**0.397**	**0.304**	**0.251**	**0.225**	0.194	0.179
OS^3VM	0.508	0.367	0.315	0.264	0.197	0.172

Table 5: The running time (sec) versus the numbers of labeled data for air pollution classification data

Algorithm	Number of labeled data					
	4	8	12	16	20	24
LS-SVM	–	0.394	0.301	0.239	**0.161**	**0.110**
S^2LS-SVM	**0.309**	**0.260**	**0.205**	**0.193**	0.177	0.155
OS^3VM	0.405	0.326	0.244	0.217	0.183	0.147

Table 6: The running time (sec) versus the numbers of labeled data for Banknote classification data

Algorithm	Number of labeled data					
	4	8	12	16	20	24
LS-SVM	–	–	0.362	0.274	**0.211**	**0.185**
S^2LS-SVM	–	**0.371**	0.334	0.319	0.305	0.282
OS^3VM	–	0.398	**0.297**	**0.268**	0.249	0.228

Each data sample records the concentrations of three most common pollutants, which are sulfur dioxide, nitrogen dioxide and PM10. At each trail of simulation, the number of anchor points and training data are set as 10 and 590, respectively. We use the remain 200 data for testing.

- The banknote dataset is a two-classes dataset obtained from UCI Machine Learning Repository [1]. This dataset contains total 1372 samples, and each data sample records some features of images that are taken from genuine and forged banknote-like specimens. At each trail of simulation, 12 data are set as anchor points, 1000 data are selected for training. Then, the remaining 360 data are used for testing.

In the simulation, the same parameter settings as the last example are used. The steady-state MCRs of different algorithms with different number of labeled data for these three datasets are summarized in Tables 1-3, respectively. From the simulation results, we can see that the MCRs of the considered three algorithms decrease as the number of labeled data increases, and the OS^3VM algorithm has the lowest MCR among these three algorithms in all the cases, which suggests that the proposed OS^3VM algorithm can be widely used in many classification problems.

We also record the running times when the MCR reduces to below 10% with different number of labeled data on the above three datasets, and the results of different algorithms are given in Tables 4-6. Note that when the labeled data are scarce, the performances of some algorithms are rather poor such that the steady-state MCRs are always larger than 10%. For these cases, we use the notation " – ". From Tables

4-6, we can find that with the increasing number of labeled data, the number of iterations for achieving a certain performance level becomes less and less, and thus the corresponding running times decrease. Besides, when the labeled data are scarce, the running time of our proposed OS^{3}VM algorithm is shorter than that of the LS-SVM, but slightly longer than that of the S^{2}LS-SVM algorithm. Yet, when the number of labeled data is more than 20, the running time of OS^{3}VM is shorter than that of the S^{2}LS-SVM, but slightly longer than that of the LS-SVM. The possible reasons for this phenomenon are illustrated as follows.

As mentioned above, the running time for achieving a certain performance level depends on two factors, the computation complexity in each iteration and the number of required iterations. As analyzed above, the OS^{3}VM need more computations in each iteration, comparing to the other two algorithms. But, it converges fastest and needs fewest iterations to achieve the required performance level. So, its running time for achieving a certain performance level can be comparable to that of the other two algorithms, and even better than that of the LS-SVM when the number of labeled data is small. As to the S^{2}LS-SVM, since it considers the effect of unlabeled data in the initial setting, the number of required iterations of S^{2}LS-SVM is close to that of OS^{3}VM when there only exist a few labeled data. But, as the number of labeled data increases, due to the negative effect brought by (over) simplification, it converges slowly. So, its running time becomes longer than that of the OS^{3}VM. For the LS-SVM, since it does not consider unlabeled data in each training processing, it converges slowly and thus requires many iterations. But, as the number of labeled data increases, the negative effect caused by ignoring unlabeled data becomes less and less, and thus the required number of iterations can be significantly reduced.

6. Conclusion

In this paper, we have proposed a new adaptive and online S^{3}VM algorithm to handle the streaming data classification problems. Compared to the S^{2}LS-SVM, the proposed OS^{3}VM fully considers the information of all the incoming data in both initial setting and latter iterations. Furthermore, considering that the standard manifold regularization brings difficulties in the updation of parameter matrix in LAEs, we have proposed a new form of manifold regularization with fixed kernel, and obtained an adaptive solution with relatively low computational complexity. Our simulation results have shown that the proposed OS^{3}VM outperforms the S^{2}LS-SVM and the LS-SVM in the classification performance for all the considered cases, especially when the labeled data are scarce. So, it can be widely applied for solving streaming data classification problems.

It is noted that although this paper focuses on the classification problem, the proposed OS^{3}VM algorithm can be easily extended to solving the online regression problems. In addition, considering that in many real-world cases, data may not be collected centrally at one node, but dispersedly collected/stored over multiple nodes, we would also like to extend the OS^{3}VM to some distributed environments and develop some distributed OS^{3}VM algorithms correspondingly.

Acknowledgment

This work was partly supported by the National Natural Science Foundation of China (Grant Nos. 61471320, 61571392 and 61631003), Zhejiang Provincial Natural Science Foundation of China (Grant No. LY17F010009), the Fundamental Research Funds for the Central Universities (Grant No. 2017QNA5009), and the National Program for Special Support of Eminent Professionals.

References

[1] C. Blake and C. Merz, UCI Repository of Machine Learning Databases, Online Available: http://archive.ics.uci.edu/ml/datasets.html.

[2] O. Chapelle, B. Scholkopf, A. Zien et al., Semi-supervised learning, MIT Press, London, UK, 2 (2006).

[3] O. Chapelle, V. Sindhwani, and S. S. Keerthi, Optimization techniques for semi-supervised support vector machines, J. Mach. Learn. Res., 9(2) (2008) 203-233.

[4] M.S. Chen, T.Y. Ho, and D.Y. Huang, Online transductive support vector machines for classification, in: Proceedings of IEEE International Conference on Information Security and Intelligence Control, 2012, pp. 258–261.

[5] B. Chen, S. Zhao, P. Zhu, and J. C. Príncipe, Quantized kernel least mean square algorithm, IEEE Trans. Neural Netw. Learn. Syst., 23(1) (2012) 22-32.

[6] B. Chen, S. Zhao, P. Zhu, and J. C. Príncipe, Quantized kernel recursive least squares algorithm, IEEE Trans. Neural Netw. Learn. Syst., 24(9) (2013) 1484-1491.

[7] C. Y. Deng, A generalization of the Sherman-Morrison-Woodbury formula, Appl. Math. Lett., 24(9) (2011) 1561-1564.

[8] G. Forestier, C. Wemmert, Semi-supervised learning using multiple clusterings with limited labeled data, Inf. Sci., 361-362(20) (2016) 48-65.

[9] G. H. Golub and C. F. Van Loan, Matrix Computation, JHU Press, 2012.

[10] W. W. Hager, Updating the inverse of a matrix, SIAM rev., 31(2) (1989) 221-239.

[11] Z. Hao, S. Yu, X. Yang, F. Zhao, R. Hu, and Y. Liang, "Online ls-svm learning for classification problems based on incremental chunk," in: Proceedings of International Symposium on Neural Networks, 2004, pp. 558–564.

[12] S. C. H. Hoi, R. Jin, J. Zhu, and M. R. Lyu, Semi-supervised SVM batch mode active learning for image retrieval, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Anchorage, AK, 2008, pp. 1–7.

[13] S. Huang and C. Li, Distributed extreme learning machine for nonlinear learning over network, Entropy, 17(2) (2015) 818-840.

[14] G. Huang, S. Song, J. N. Gupta, and C. Wu, Semi-supervised and unsupervised extreme learning machines, IEEE Trans. cybern., 44(12) (2014) 2405-2417.

[15] J. Li, Y. Cao, Y. Wang, and H. Xiao, "Online learning algorithms for double-weighted least squares twin bounded support vector machines," Neural Process. Lett., 45(1) (2016) 1-21.

[16] C. L. Liu, W. H. Hsaio, C. H. Lee, T. H. Chang, and T. H. Kuo, "Semi-supervised text classification with universum learning." IEEE Trans. Cybern., 46(2) (2017) 462-473.

[17] A. Mousavi, S. S. Ghidary, and Z. Karimi, "Semi-supervised intrusion detection via online laplacian twin support vector machine," in: Proceedings of Signal Processing and Intelligent Systems Conference, 2015, pp. 138–142.

[18] S. Nan, L. Sun, B. Chen, Z. Lin, and K. A. Toh, "Density-dependent quantized least squares support vector machine for large data sets," IEEE Trans. Neural Netw. Learn. Syst., 28(1) (2017) 94-106.

[19] S. A. Nene, S. K. Nayar, H. Murase et al., Columbia object image library (coil-20), Tech. Rep., 1996.

[20] H. Ohlsson and L. Ljung, Semi-supervised regression and system identification, Three Decades of Progress in Control Science, Springer, 2010, 343-360.

[21] B. Schölkopf and A. J. Smola, Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond, MIT Press, Cambridge, MA, 2002.

[22] P. Shen and C. Li, Distributed information theoretic clustering, IEEE Trans. Signal Process., 62(13) (2014) 3442-3453.

[23] K. Taghipour and H. T. Ng, "Semi-supervised word sense disambiguation using word embeddings in general and specific domains," in: Proceedings of Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2015, pp. 314–323.

[24] E. Tu, Y. Zhang, L. Zhu, J. Yang, and N. Kasabov A graph-based semi-supervised k nearest-neighbor method for nonlinear manifold distributed data classification, Inf. Sci., 367-368(1) (2016) 673-688.

[25] S. Xu, X. An, X. Qiao, L. Zhu, and L. Li, Semi-supervised least-squares support vector regression machines, J. Inf. Comput. Sci., 8(6) (2011) 885-892.

[26] X. Yang, L. Tan, and L. He, "A robust least squares support vector machine for regression and classification with noise," Neurocomputing, 140 (2014) 41-52.

[27] J. Yoo and H. J. Kim, Online estimation using semi-supervised least square SVR, in: Proceedings of IEEE International Conference on System, Man and Cybernetics, San Diego, CA, 2014, pp. 1624–1629.

[28] Z. Zhang, L. Zhang, M. Zhao, W. Jiang, F. Li, and F. Li, "Semi-supervised image classification by nonnegative sparse neighborhood propagation," in: Proceedings of ACM International Conference on Multimedia Retrieval, 2015, pp. 139–146.

[29] M. Zhao, C. Zhan, Z. Wu, and P. Tang, "Semi-supervised image classification based on local and global regression," IEEE Signal Process. Lett., 22(10) (2015) 1666-1670.

[30] Z.-H. Zhou and M. Li, Semisupervised regression with cotraining-style algorithms, IEEE Trans. Knowl. Data Eng., 19(11) (2007) 1479-1493.